# Отчет по лабораторной работе №5 по курсу «Функциональное программирование»

Студент группы М8О-307Б-19 Калугин Кирилл Алексеевич, № по списку 11.

Контакты: netter2@rambler.ru  
Работа выполнена: 17.05.2022  
Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806  
Отчет сдан:   
Итоговая оценка:   
Подпись преподавателя:

### 1. Тема работы

Обобщённые функции, методы и классы объектов.

### 2. Цель работы

### Научиться определять простейшие классы, порождать экземпляры классов, считывать и изменять значения слотов, научиться определять обобщённые функции и методы.

### 3. Задание (вариант № 5.23)

Задание: Определить обобщённую функцию и методы on-single-line3-p - предикат,

* принимающий в качестве аргументов три точки (радиус-вектора) и необязательный параметр tolerance (допуск),
* возвращающий T, если три указанные точки лежат на одной прямой (вычислять с допустимым отклонением tolerance).

Точки могут быть заданы как декартовыми координатами (экземплярами cart), так и полярными (экземплярами polar).

### 4. Оборудование студента

Персональный компьютер, процессор Intel® Core™ i5-9400F CPU 2.90GHz 2.90GHz, память 16ГБ, 64-разрядная система.

### 5. Программное обеспечение

ОС Windows 10, компилятор Steel Bank Common Lisp 2.2.2 (x86), текстовый редактор Atom 1.60.

### 6. Идея, метод, алгоритм

Для выполнения задания была создана общая функция «main». У нее есть два метода – один для декартовых координат, второй – для полярных. Сами классы координат также определены. Кроме того, реализованы две вспомогательные функции – «to-cart» и «l-belong». Первая - преобразует координаты в декартовы, вторая - проверяет – проходит ли линия, проведенная через точки №1 и №2 через окрестность размера tolerance точки №3. При выполнении функции «main» происходит вызов функции «l-belong» от всех трех комбинаций точек. Если хотя бы в одном случае было получено “T” – значит возможно провести прямую, проходящую через 2 точки и окрестность третьей. Такой подход не идеален и имеет случай, в котором программа выдает неверный ответ (подробнее – см. п. 10).

### 7. Сценарий выполнения работы

### 8. Распечатка программы и её результаты

**Программа**

(defclass cart ()

((x :initarg :x :reader cart-x)

(y :initarg :y :reader cart-y)))

(defclass polar ()

((radius :initarg :radius :accessor radius)

(angle :initarg :angle :accessor angle))

)

(defmethod cart-x ((p polar))

(float (\* (radius p) (cos (angle p))) 1.0)

)

(defmethod cart-y ((p polar))

(float (\* (radius p) (sin (angle p))) 1.0)

)

(defgeneric to-cart (arg)

(:method ((c cart))

c

)

(:method ((p polar))

(make-instance 'cart

:x (cart-x p)

:y (cart-y p)

)

)

)

(defun l-belong (point1 point2 point3 &optional (tolerance 0))

(>= tolerance (/ (abs (+ (\* (- (cart-y point2) (cart-y point1)) (cart-x point3))

(\* (- (cart-x point1) (cart-x point2)) (cart-y point3))

(\* (cart-x point2) (cart-y point1))

(\* (- 0 1) (cart-y point2) (cart-x point1))))

(sqrt (+ (\* (- (cart-y point2) (cart-y point1)) (- (cart-y point2) (cart-y point1)))

(\* (- (cart-x point2) (cart-x point1)) (- (cart-x point2) (cart-x point1)))))))

)

(defgeneric main (v1 v2 v3 &optional tolerance)

(:method ((c1 cart) (c2 cart) (c3 cart) &optional (tolerance 0))

(OR (l-belong c1 c2 c3 tolerance) (l-belong c1 c3 c2 tolerance) (l-belong c2 c3 c1 tolerance))

)

(:method ((p1 polar) (p2 polar) (p3 polar) &optional (tolerance 0))

(OR (l-belong (to-cart p1) (to-cart p2) (to-cart p3) tolerance) (l-belong (to-cart p1) (to-cart p3) (to-cart p2) tolerance) (l-belong (to-cart p2) (to-cart p3) (to-cart p1) tolerance))

)

)

;(main (make-instance 'polar :radius 10 :angle ( - (/ (\* 3 pi) 4))) (make-instance 'polar :radius 1 :angle ( - (/ (\* 6 pi) 8))) (make-instance 'polar :radius 3 :angle (/ pi 4)))

;(main (make-instance 'cart :x 1 :y 1) (make-instance 'cart :x 2 :y 2) (make-instance 'cart :x 4 :y 3))

;(main (make-instance 'cart :x 1 :y 1) (make-instance 'cart :x 2 :y 2) (make-instance 'cart :x 4 :y 3) 1)

**Результаты**

CL-USER> (main (make-instance 'polar :radius 10 :angle ( - (/ (\* 3 pi) 4))) (make-instance 'polar :radius 1 :angle ( - (/ (\* 6 pi) 8))) (make-instance 'polar :radius 3 :angle (/ pi 4)))

T

CL-USER> (main (make-instance 'cart :x 1 :y 1) (make-instance 'cart :x 2 :y 2) (make-instance 'cart :x 4 :y 3))

NIL

CL-USER> (main (make-instance 'cart :x 1 :y 1) (make-instance 'cart :x 2 :y 2) (make-instance 'cart :x 4 :y 3) 1)

T

### 9. Дневник отладки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Дата, время | Событие | Действие по исправлению | Примечание |
| 1 |  |  |  |  |

### 10. Замечания автора по существу работы

Как было сказано в п.6, алгоритм решения неидеален. Он не находит решения в случаях, когда прямая проходит лишь через одну точку и две окрестности или три окрестности. Примером может послужить набор точек (1, 1) (2, 2) (0, 3) с заданным допуском равным 1.

В связи с этим я ввел ограничение – итоговая прямая должна проходить хотя бы через 2 точки.

### 11. Выводы

В данной лабораторной работе я узнал об обобщенных функциях, методах и классах в Common Lisp и написал программу, использующую их. Программа работает правильно и прошла все тесты.